

# Evolutionary Algorithms with Dissortative Mating on Static and Dynamic Environments

Carlos M. Fernandes<sup>1,2</sup> and Agostinho C. Rosa<sup>1</sup>

<sup>1</sup>*Laseeb – Instituto de Sistemas e Robótica – Instituto Superior Técnico*

<sup>2</sup>*Depart. de Arquitectura y Tecnología de Computadores – University of Granada*

<sup>1</sup>*Portugal*

<sup>2</sup>*Spain*

## 1. Introduction

Evolutionary Algorithms (EAs) (Bäck, 1996) mimic the process of natural selection by recombining the most promising solutions to a problem from a population of individuals, each one representing a possible solution. There are several methods to select the individuals, but all of them follow the same general rule: good (or partially good) solutions must be chosen more often for recombination events than poorer solutions. In traditional Genetic Algorithms (GAs), for instance, the chromosomes are recombined via a crossover operator over a certain number of generations until a stop criterion is reached. The parents are selected according to their fitness values, that is, better solutions have larger probability to be chosen to generate offspring. By considering merely the quality of solutions represented in the chromosomes when selecting individuals for mating purposes, the traditional GAs emulate what, in nature, is called *random mating* (Roughgarden, 1979; Russel, 1998), that is, mating chance is independent of genotypic or phenotypic distance between individuals.

However, random mating is not the sole mechanism of sexual reproduction observed in nature. *Non-random mating*, which encloses different kinds of strategies based on parenthood or likeness of the agents involved in the reproduction game, is frequently found in natural species, and it is believed to be predominant among vertebrates. Humans, for instance, mate preferentially outside their family tree: this non-random mating scheme is called *outbreeding* and has its opposite in *inbreeding*, a selection strategy where individuals mate preferentially with their relatives (Roughgarden, 1979; Russel, 1998). It is often stated that inbreeding decreases the genetic diversity in a population while outbreeding increases that same diversity (Russel, 1998). In addition, inbreeding will increase the normal rate of a harmful allele present in the family. If inbreeding is extensive and intensive, homozygosity will increase in frequency and the family experiences a growth in the genetic load (measure of all of the harmful recessive alleles in a population or family line) of the harmful allele.

*Assortative mating* is another non-random mating mechanism, in which individuals choose their mates according to phenotypic similarities (Roughgarden, 1979; Russel, 1998). When similar individuals mate more often than expected by chance, we are in presence of positive assortative mating (or assortative mating in the strict sense). When dissimilar individuals

mate more often, the scheme is called negative assortative mating (or *dissortative mating*). In humans, assortative mating is well exemplified by the correlation between heights or intelligence in partners. On the other hand, humans do not mate assortatively with respect to blood groups. This kind of behavior, which selects assortatively for some traits and not others, makes it difficult to unmask the effects of assortative mating in the population. In fact, human assortative mating is not completely positive except for some small and isolated communities (the Old Order Amish, for instance).

Positive assortative mating results in an average increase in homozygosity and in an increase in population variance. However, this does not mean that genetic diversity is increasing. In fact, this type of mating may result in highly distinct cluster of similar genotypes, thus playing a crucial role when speciation without geographic barriers occurs (sympatric speciation) (Todd & Miller, 1991). Dissortative mating, on the other hand, has the primary consequence of a progressive increase in the frequency of heterozygous genotypes; the increase in the diversity of the population is a direct consequence of these changes in the genotype frequencies. Evidences show that mating is very unlikely to be random in nature and may have the potential to act as an evolutionary agent, although its effects are very complex and hard to model and analyze (Jaffe, 1999). Even so, artificial life models presented by Jaffe (1999) and Ochoa et al. (1999) shed some light into the subject, and gave empirical support to the hypothesis that mating is not likely to be random in nature and that assortative and dissortative mating may produce higher survival rates among individuals evolving in, static and dynamic environment, respectively. While in dynamic landscapes genetic variability is fundamental to a quick and effective response to changes, in static environments diversity is not so important. In fact, natural organisms move towards an optimal degree of genetic variability that depends on the environment, via some mating scheme. Environment itself appears to guide the evolution of mating strategies.

In *Evolutionary Computation* (Bäck, 1996), selective pressure and genetic diversity are two major topics, probably those of primary importance (Whitley, 1988). Pressure and diversity are closely related to the delicate equilibrium between exploration and exploitation needed in order to have "safe" search in EAs. Therefore, non-random mating naturally came out in EAs research field in order to deal with the problem of genetic diversity and premature convergence: some efficient algorithms appeared, especially when applied to problems where the genetic diversity is needed in order to maintain exploration high and avoid local optima traps. In addition, diverse search stages usually call for different balance between exploration and exploitation mechanisms. To an initial strong explorative stage, the algorithm gradually must enter a more exploitive phase, where the neighborhood of good solutions found so far is inspected in order to reach the global optimum. When the problem's environment change over time, that is, when dealing with dynamic optimization, genetic diversity becomes even more important, since full convergence must be avoided: the algorithm must maintain sufficient diversity to readapt itself to a change in the fitness function, even if it has converged to the current optimum. In dynamic environments, it is often more important to track the best solution than to converge, that is, it may be sufficient to keep the population near the optimum, even if returning only near-optimal solution, thus avoiding the risk of a full convergence in a specific period of the search, which would reduce the possibilities of readaptation after a change.

Very often recombination is associated with exploitation while mutation is said to play a determinant role in exploration by preventing alleles becoming extinct. While this appears

to be true, it may have misled some researches towards assortative mating instead of dissortative, because of the higher exploitation performed by the first strategy. If similar individuals tend to mate, it is more likely that their neighboring space is closely inspected. On the other hand, several studies on dissortative mating showed empirical evidence that this scheme is more adapted to a wide range of problems, both static and dynamic (Craighurst & Martin, 1995; Eschelmann, 1991; Eschelmann & Schaffer, 1991; Fernandes et al., 2000, 2001; Fernandes & Rosa 2001; Fernandes, 2002; García-Martínez et al., 2007; Matsui, 1999; Ochoa et al., 2005) – see next section for a state-of-the-art review.

This chapter proposes a review and an empirical study on EAs with dissortative mating strategies and their application to static and dynamic problems. Dissortative mating will be discussed within a biological framework and some Artificial Life models will be analyzed; a detailed description of several methods found in EAs literature will be also given. The empirical study will be centered on the Variable Dissortative Mating GA (VDMGA), which was recently presented in (Fernandes & Rosa, 2008) by the authors of this chapter. VDMGA holds a mechanism that varies GA's mating restrictions during the run, by means of a simple rule based on the number of chromosomes created in each generation and indirectly influenced by the genetic diversity of the population. The empirical study presented in (Fernandes & Rosa, 2008) shows that VDMGA performs well when applied to a wide range of problems: it consistently outperforms traditional GAs and assortative mating GAs, and it is faster and more robust than some previously proposed dissortative mating GAs. Results suggest that VDMGA's ability to escape local optima and converge more often to the global solution may come from maintaining the genetic diversity at a higher level when compared with traditional GAs. VDMGA's genetic diversity naturally leads the research towards the application of the algorithm on Dynamic Optimization Problems (DOPs). Due to their specific characteristics, DOPs require additional tools, many of them different from those widely studied by EAs researchers on static problems. Memory schemes and niching (Branke & Schmeck, 2002) are some of the techniques used to tackle DOPs. Strategies for maintaining genetic diversity and/or introducing novelty in the EAs populations are also very efficient strategies when solving dynamic problems (Branke & Schmeck, 2002). In this chapter, the original VDMGA is subject to minor modifications, and then applied to DOPs benchmarks and compared to other GAs. The results confirm the predictions and show that VDMGA may improve other GAs' performance on changing environments. As already been observed when tackling static fitness functions (Fernandes & Rosa, 2008), dissortative mating, via a simple and easily tunable algorithm with diversity preservation, reveals interesting skills when evolving in dynamic environments.

## 2. Non-random mating evolutionary algorithms

This section describes some EAs with outbreeding, assortative and dissortative mating strategies found in the literature. A special emphasis is given to the ones that, to the extent of the authors of this chapter knowledge, were seminal in their line of work, and to those that preceded (or are, at some level, related to) VDMGA.

In the GA with outbreeding described in (Craighurst, 1995), individuals with a certain degree of parenthood are not allowed to recombine and generate offspring. An incest prevention degree is defined in the beginning of the run and remains unchanged until the convergence criterion is fulfilled. This degree defines how far back in the family tree of an individual the GA must inspect in order to prevent the recombination events. This policy

does not completely restrict mating between similar individuals, but it sure decreases its frequency since related individuals tend to share a large amount of common alleles. Tests (Craighurst, 1995) compare the outbreeding GA with a standard GA when applied to the Traveling Salesman Problem. The non-random mating algorithm outperformed the standard GA but the differences in the algorithms' performances were noticed mainly with low mutation rates. This is not surprising since incest prohibition is supposed to maintain the genetic diversity of the population at a higher level for longer periods, thus reducing the need for mutation to introduce genetic novelty into converging populations. Fernandes et al. (2000) combined the outbreeding strategy proposed in (Craighurst, 1995) with a varying population size GA (Arabas, 1994) to create the *non-incest Genetic Algorithm with Varying Population Size* (niGAVaPS). The results showed that the two mechanisms worked together well in order to find the optimum of Four Peaks and Royal Road R4 functions. Tests made with the algorithm ranging through different degrees of incest prohibition showed improvements in the capability of escaping local optima when the individuals are not allowed to mate with their parents and siblings.

There are several studies indicating that dissortative mating may improve EAs performance by maintaining the genetic diversity of the population at a higher level during the search process. For instance, CHC (Eschelman, 1991; Eschelman & Schaffer, 1991) which stands for *Cross generational elitist selection, Heterogeneous Recombination and Cataclysmic Mutation*, is a variation of the standard GA that holds a simple mechanism of dissortative mating which has given proofs of being rather effective in a wide range of problems. Although the title in (Eschelman & Schaffer, 1991) may suggest that CHC is an outbreeding GA, a closer look reveal that the algorithm uses a dissortative mating strategy in order to prevent premature convergence. CHC uses no mutation in the classical sense of the concept, but instead it goes through a process of macro-mutation (or hyper-mutation) when the best fitness of the population does not change after a certain number of generations. The genetic diversity is assured by a highly disruptive crossover operator, the Half Uniform Crossover (HUX) (Eschelman & Schaffer, 1991), and a reproduction restriction that assures that selected pairs of chromosomes will not generate offspring unless their Hamming Distance is above a certain threshold. CHC search process goes as follows. In each generation,  $p/2$  pairs of chromosomes are randomly selected from the population with size  $p$ . All pairs are submitted to the reproduction process. First, their Hamming distance is computed. If the value is found to be above the threshold then the chromosomes generate two children with the HUX operator. When the process is concluded, the newly generated population of  $p'$  offspring replaces the worst chromosomes in the main population, therefore maintaining the size of the population. The threshold is usually set in the beginning of the runs to  $1/4$  of the chromosome length, and decremented when no offspring is generated. When the algorithm is stuck in local optima, a cataclysmic mutation is applied by replacing the entire population, except the best chromosome, with mutated copies of that individual.

The *Assortative Mating GA* (AMGA) was introduced in (Fernandes et al., 2001). The only difference between AMGA and a standard GA is the way parents are selected for recombination. In each recombination event one parent (*first parent*) is select by any traditional method. Then, a set of  $n$  individuals is selected by the same method. After computing the similarity between the first parent and all the  $n$  individuals in the set, the second parent is chosen according to the type of assortative mating in progress. If the algorithm is the *positive Assortative Mating GA* (pAMGA) the individual more similar to the

first parent is chosen. With the *negative Assortative Mating* GA (nAMGA) the individual less similar is chosen as the second parent (please remember that negative assortative is the same as dissortative). The intensity of the non-random mating scheme may be controlled by the size of the set of candidates to the second parent position. Increasing  $n$  increases the frequency of mating between dissimilar (if negative assortative) or similar (if positive) individuals. Experiments with the algorithm solving a vector quantization problem showed pAMGA and standard GA performed similarly, while nAMGA outperformed both (Fernandes et al., 2001). Increasing the size of the candidates set resulted in higher success rates (number of runs in which the global optima was found) of nAMGA. In (Fernandes & Rosa, 2001), the algorithm was combined with a varying population size mechanism, tested with a Royal Road (R4) function (Mitchell, 1994) and compared with a standard GA and the niGAVaPS (Fernandes et al., 2000). The negative assortative mating (or dissortative mating) strategy has proven to be more able in escaping Royal Road's local optima traps. pAMGA was also tested under the same conditions but its performance was clearly inferior to standard GA.

A similar idea was tested by Ochoa et al. (2005) on dynamic environments. The authors tested haploid and diploid GAs with assortative mating (where parents are selected as in AMGA) on a knapsack problem with moving extrema, and nAMGA was more able to track dynamic optima. Standard GA often failed to track the optima but the worst performance was attained by pAMGA. In general, the haploid algorithms produced better results than the diploid ones. The authors also discuss the optimal mutation rate for different strategies. By means of exhaustive tests, they concluded that the optimal mutation rate increases when the mating strategy goes from negative (dissortative) to positive assortative. These results were predictable: dissortative mating is supposed to maintain the population diversity at a higher level, reducing the amount of mutation needed in order to prevent the premature convergence of the population. In this line of work, the same authors proposed a study on the error threshold of replication in GAs with different mating strategies (Ochoa, 2006; Ochoa & Jaffe, 2006). The error threshold is a critical mutation rate beyond which structures obtained by an evolutionary process are destroyed more frequently than selection can reproduce them. By evolving a GA on four different fitness landscapes, the authors first conclude that recombination shifts the error threshold toward lower values. Then, the tests show that assortative mating overcomes this effect by increasing the error threshold, while the dissortative strategy pushes the error into lower values. The authors argue that this study may have effects on both natural and artificial systems since it supports the hypothesis that assortative mating overcomes some of the disadvantages inherent to sex. They also intend to shed some light into the relation between mutation rates and mating strategies in EAs. This last issue is directly related with the idea that assortative mating increases the optimal mutation rate of an EA, while dissortative strategies decreases it. This behavior has already been observed in (Fernandes, 2002) and (Ochoa et al., 2005).

Fernandes & Rosa (2006) proposed the *Self-Regulated Evolutionary Algorithm* (SRPEA). SRPEA is an algorithm with a dynamic on-the-fly variation of the population size. Selected individuals are recombined to generate offspring only if their Hamming distance is above a threshold value. That value changes over time, depending on the number of newborn individuals and deaths in each generation. Individuals die (that is, are removed from the population) only when their lifetime (which is set to specific value in the beginning of the search depending on the individual's fitness) reaches zero, which means that parents and

children may belong to the same population. An empirical study demonstrated that the algorithm self-regulates its population size: there are neither uncontrolled demographic explosions nor quasi-extinction long stages, as it is observed in the dynamics of other varying population EAs (Arabas, 1994). VDMGA, the main algorithm in this chapter's study, is directly related to SRPEA.

In (García-Martínez et al., 2006), an assortative mating strategy is used to implement a local search genetic algorithm. The approach is consistent with the fact that crossover is the main mechanism of a GA generating local search, and assortative mating, by its own characteristics, tends to increase the strength of exploitation, thus leading to a more intensive local search. On the other hand, García-Martínez et al. (2007) introduced a real-coded genetic algorithm with disassortative mating. The authors show that the inclusion of that mating strategy increases the performance of the GA on a set of proposed problems. In addition, empirical analysis indicates that the merits of disassortative mating are clearer with lower values of  $a$  parameter of the PBX- $a$  crossover (Lozano et al., 2004). This observation is closely related with the optimal mutation rate issue described above, since  $a$  determines the spread of the probability distribution used to create offspring with PBX- $a$ . This way, parameter  $a$  acts as genetic diversity controller, with higher values leading to GAs with higher exploratory capabilities, as it happens with mutation rate values. Therefore, if disassortative mating is expected to decrease optimal mutation rates, optimal values of  $a$  may also be dependent on the mating strategy chosen for the GA, being lower when dissimilar individuals have more chance to generate offspring.

A large number of other GAs with non-random mating may be found in Evolutionary Computation literature. A few are briefly described in the following paragraph.

Mauldin (1984) proposed a method to avoid similar individuals in the population based on a Hamming distance restriction. CHC is in some way a descendent of Mauldin's method, and, as a result, so is VDMGA. Hillis (1992) described a co-evolutionary computation paradigm with assortative mating applied to a sorting network problem. The author does not provide results comparing the proposed strategy and random mating but it states that the choice on assortative mating was inspired by some problem characteristics rather than genetic diversity concerns. Ronald (1995) introduced the concept of seduction in GAs, which consists in selecting the second parent according to the preferences of the first parent. After the first chromosome involved in a recombination event is selected, all other individuals in the population are provided with a secondary fitness according to certain rules that reflects the preferences of the first parent. Then, the second parent is chosen according to the secondary fitness. Petrowski proposes (1997) speciation in order to restrict mating. De et al. (1998) proposed genotypic and phenotypic assortative mating. The new approaches are compared with standard GA and CHC on some well-known test functions and on the problem of selecting the optimal set of weights in a multilayer *perceptron*. Phenotypic assortative mating revealed to be the best strategy, outperforming standard GA and CHC on the range of proposed problems. Matsui (1999) incorporated disassortative mating within the tournament selection strategy. After the first parent is selected, the second parent is chosen according to a function that depends on the individual fitness and the Hamming distance to the first parent (all individuals in the population are inspected in order to determine the distance to the first parent). In addition, the author incorporates a family-based selection mechanism that, by applying selection and replacement at family level (two parents and two offspring), maintains the genetic diversity of the population. Ting et al. (2003) introduced

the Tabu Genetic Algorithm (TGA). TGA combines the characteristics of GAs and Tabu Search (Glover, 1986), by incorporating a taboo list in a traditional GA that prevents inbreeding and maintains genetic diversity. An aspiration criterion is also used by TGA in order to allow some crossovers even if they violate the taboo. Since incest prevention efficiency is sensitive to mutation rate, the authors include a self-adaptive mutation in TGA. The process is somehow similar to the cataclysmic mutation that occurs in CHC, since mutation in TGA occurs in presence of a deadlock situation, that is, when the genetic diversity of the population as decreased down to a level were allowed recombination is almost or even impossible to occur. Finally, Wagner & Affenzeller (2005) introduced the SexualGA, which simulates sexual selection within the frame of a GA and uses two different selection schemes in the same population.

### 3. Dynamic optimization problems

A problem is said to be a Dynamic Optimization Problem (DOP) when there is a change in the fitness function, problem instance or restrictions, thus making the optimum change as well. When changes occur, solutions already found may be no longer valuable and the process must engage in a new search effort. Traditional EAs, for instance, may encounter some difficulties while solving dynamic problems: if the first convergence stage reduces population diversity, then the algorithm may not be able to react to sudden changes. The crucial and delicate equilibrium needed between exploration and exploitation in static environments becomes even more important and complex when dealing with DOPs. In addition, if the change is detectable (which not always possible), it is hard to decide if it is better to continue the search with same population, after a shift in the environment, or if a restart is more efficient. The extent of the change is of crucial importance in that decision. This problem was stated by Branke & Schmek (2002), which suggested a classification of DOPs and a classification of the most widespread EAs that deal with changing environments. One standard approach to deal with DOPs is to regard each change as the arrival of a new optimization problem that has to be solved from scratch. However, this simple approach is often impractical since solving a problem from scratch without reusing information from the past might be time consuming, a change might not be identifiable directly, or the solution to the new problem should not differ too much from the solution of the old problem. Thus, as in the on-line tracking process suggested in (Angeline, 1997), it has been recommended in (Branke, 1999; Branke, 2002; Branke & Schmeck 2002) to have an optimization algorithm that is capable of continuously adapting the solution to a changing environment, reusing the information gained in the past. Since natural adaptation is a continuous and continuing process and EAs have much in common with natural evolution, they seem to be a suitable candidate for this task. However, evolutionary approaches typically converge to an optimum and thereby lose the diversity necessary for efficiently exploring the search space and consequently also the ability to adapt to a change in the environment (Branke, 2002; Branke & Schmeck 2002). The problem here can be stated as seeking an appropriate balance between two contradictory characters of the search procedure, those between the exploring (ideal for gathering new solutions) and exploiting (making the best use of past solutions) nature of the algorithm. Over the past few years, a number of authors have addressed the problem of convergence and subsequent loss of adaptability in many different ways. According to (Branke e Schmeck 2002), most of these approaches could be grouped into one of the following three categories established by them:

1. React on Changes: The EA is run in standard fashion, but as soon as a change in the environment is detected, explicit actions are taken to increase diversity and thus facilitating the shift to the new optimum.
2. Maintaining Diversity throughout the run: Convergence is avoided all the time and it is hoped that a spread-out population can adapt to changes more easily.
3. Memory-based Approaches: The EA is supplied with a memory to recall useful information from past generations, which seems especially useful when the optimum repeatedly returns to previous locations.

Techniques such as *Hypermutation* (Cobb, 1990) pursue the first category, keeping the whole population after a change but increasing population diversity by drastically increasing the mutation rate for some number of generations. Please note that reacting to changes assumes that changes are detectable, a condition, as already stated, that is not always fulfilled (Branke, 2002).

The *Random Immigrants Genetic Algorithm* (RIGA) (Grefenstette, 1992) is an example of a strategy that falls in the second category. In RIGA the population is partly replaced by  $r_r$  randomly generated individuals in every generation. This guarantees the introduction of new genetic material in every time step and avoids the convergence of the whole population to a narrow region of the search space. The performance is affected by the parameter  $r_r$ . RIGA is used in the following sections to evaluate VDMGA's performance on DOPs; therefore, its pseudo-code is presented here:

*Algorithm 1: Random Immigrants Genetic Algorithm*

```

initialize Population(P)
evaluate Population(P)
while (not termination condition) do
    P ← Replace Fraction of Population (P,  $r_r$ )
    create P.new by selection, crossover and mutation of P
    P ← P.new
end while

```

The following algorithms may also be classified in category 2. As described in the previous section, the *negative Assortative Mating Genetic Algorithm* (nAMGA) (Fernandes & Rosa 2001) is used in (Ochoa et al., 2005) to solve a knapsack DOP. Negative assortative mating (or dissortative mating), by preventing the recombination of similar individuals, slows down the expected diversity loss of traditional GAs thus having the proper characteristics to be classified within category 2. The *co-evolutionary agent based model of genotype editing* (ABMGE) (Huang et al., 2007) use several genetic editing characteristics that are gleaned from the RNA editing system as observed in several organisms. Their results outperformed traditional EAs via obtaining greater phenotypic plasticity. In (Tinós & Yang, 2007), a RIGA associated with the Bak-Sneppen model is presented and tested on DOPs: the *Self-Organized Random Immigrants Genetic Algorithm* (SORIGA). Bak-Sneppen (Bak & Sneppen, 1993) is known as a Self-Organized Critically model, a phenomenon that was detected in 1987 by Bak, Tang and Wiesenfeld (Bak et al., 1987), and which characterized by displaying scale invariant behavior. When associated with EAs it may periodically insert large amounts of new material in the population or completely reorganize a solution to a problem. For those reasons, it soon was adopted by EA researchers in order to provide new means to control parameter values or maintain population diversity, thus avoiding premature convergence to

local optima. DOPs research field was a logical following step. Besides SORIGA, another approach has been recently proposed by Fernandes et al. (2008a), in which the Sandpile model (Bak et al., 1987) is attached to a GA in order to solve DOPs.

Another kind of approach is to supply the algorithm with some sort of memory, storing good partial solutions in order to reuse them later (category 3). This can be advantageous in cases where the environment is changing periodically, and repeated situations occur. However, they also could be counterproductive if the environment changes dramatically with open-ended novelty. Memory may be provided in two general ways: *implicitly* by using redundant representations, or *explicitly* by introducing an extra memory and formulating strategies to deposit and retrieve solutions later. Generally, the most prominent approach to implicit memory and redundant representation is multiploidy (Goldberg & Smith, 1987). On the other hand, while redundant representations allow the EA to implicitly store some useful information during the run, it is not clear that the algorithm actually uses this memory in an efficient way. As an alternative, some approaches use an explicit memory in which specific information is stored and reintroduced into the population at later generations, as in (Louis & Xu, 1996). Branke (1999) compared a number of replacement strategies for inserting new individuals into a memory stressing the importance of diversity for memory-based approaches.

Estimation of Distribution Algorithms (EDAs) (Pelikan, Goldberg & Lobo, 1999; Lorrãña & Lozano, 2002) is a class of EAs where a probability model replaces an explicit representation of the population. In the last decade, research on EDAs has experienced a continuous and consistent growth. However, only recently the DOP issue has started to raise a strong interest on EDAs' researchers. For instance, the Population Based Incremental Learning (PBIL) (Baluja, 1994) - one of the first EDAs - is used in (Yang & Xiao, 2005) to solve DOPs created by a problem generator proposed by the same authors. The authors compare several versions of PBIL with GAs and RIGAs. In (Yang, 2005), the author proposes the Univariate Marginal Distribution Algorithm (UMDA) with enhanced memory and the results of the experiments show that the memory is efficient in dynamic environments. In addition, a combination of memory and random immigrants for the UMDA is studied. Lima et al. (2008) investigates the incorporation of restricted tournament replacement (RTR) in the extended compact genetic algorithm (ECGA) (Harik et al., 1999) for solving problems with non-stationary optima. (RTR is a simple yet efficient niching method used to maintain diversity in a population of individuals.) Finally, Fernandes et al. (2008) proposed a new update strategy for UMDA based on Swarm Intelligence.

Some recent proposals have been made using a Swarm Intelligence (Bonabeau, Dorigo & Theraulaz, 1999) approach to attempt to solve dynamic problems. Swarm Intelligence is the property of a system whereby the collective behaviors of simple entities interacting locally with their environment cause global patterns to emerge. In (Guntsch & Middendorf, 2002) the authors applied population based ACO algorithms for tracking extrema in dynamic environments. Others, like (Ramos et al., 2005) developed distributed pheromone layering over the dynamic environment itself, in order to track different peaks. Finally, Fernandes et al. (2007) developed the *Binary Ant Algorithm* (BAA), based on the ACO framework, to take advantage of ACO's ability to solve combinatorial DOPs and generalize it to binary DOPs. However, BAA may also be regarded as a kind of EDA, since, like this class of algorithms, BAA creates the possible solutions to a problem via a transition probability model. Actually,

there have been recent attempts to unify ACO and EDAs into the same framework (Zlochin, et al., 2004).

#### 4. The variable dissortative mating genetic algorithm

To model dissortative mating in EAs, some kind of relaxation policy may be needed in order to avoid a freezing population, since evolution eventually leads the search process into a stage of low diversity, where all the individuals are almost identical. In addition, the population usually searches for an optimal degree of genetic variability according to the landscape were it evolves. It is possible that the population movement towards the optimal regions of the landscape also requires different levels of genetic diversity along the way, in order to maintain a robust search. Therefore, the degree of assortative or dissortative mating should vary along the run in order to deal with the inevitable decrease in diversity and to follow the search path of the population. Some methods try to maintain the diversity in a permanent high level, but that may be incompatible with the desirable convergence of the algorithm. For instance, a constant macro-mutation certainly maintains the diversity of the population, but the expected success of an EA based on such premises is not high. Diversity by itself is not a guarantee of a successful search through the landscape.

The *Variable Dissortative Mating Genetic Algorithm* (VDMGA) (Fernandes & Rosa, 2008) is a non-random mating GA, which incorporates an adaptive Hamming distance mating restriction that tends to relax as the search process advances, but may be occasionally reinforced. The algorithm works in the following way. When the first population is randomly created, a threshold value is set to an initial level equal to  $L-1$ , where  $L$  is the chromosome length. Then, offspring may be created by selecting pairs of parents (by any method), followed by recombination and mutation. However, recombination only occurs if the genetic distance (Hamming distance in implementation made for this chapter) between the two parents is found to be above the threshold. If not, the recombination event is considered as "failed" and another pair of chromosomes is selected until  $N/2$  pairs have tried to recombine (where  $N$  is the size of the population). When this process ends, the amount of successful and failed recombination events is compared, and the threshold is incremented if successful mating exceeds failed mating. Otherwise, threshold is decremented (the process repeats if no mating succeeded). This way, the threshold is indirectly controlled by the diversity of the population. After the reproduction cycle is completed, a new population is created by selecting the  $N$  best members from the parents' population and newly generated offspring (if a parent and a child have the same fitness then the child is chosen). Parents and children compete together for survival, conducting to a highly selective algorithm (VDMGA belongs to the class of steady-state GAs). The process repeats until a stop criterion is reached.

VDMGA's threshold value evolves in conformity with the genetic diversity of the population. When diversity decreases, threshold tends to be decremented since the frequency of unsuccessful mating will necessarily increase. However, the mutation operator introduces some variability in the population which may result in occasional increments of the threshold that moves it away from zero (if threshold reaches zero, all individuals are allowed to crossover, like in random mating GAs). Tests performed on several functions confirmed this predicted behavior (Fernandes & Rosa, 2008).

Two changes must be made on the original VDMGA presented in (Fernandes & Rosa, 2008) in order to solve DOPs with an enhanced performance.

## Algorithm 2: Variable Dissortative Mating Genetic Algorithm

```

initialize Population(P) with  $size(\mathbf{P}) = N$ 
evaluate Population(P)
set initial threshold( $iT$ )           /*  $iT \leftarrow L-1$  for static problems;  $iT \leftarrow L/4$  for DOPs*/
threshold(T)  $\leftarrow iT$ 
  while (not termination condition)
    create new individuals P.new
    evaluate new individuals P.new
    if (static problem)
      P  $\leftarrow \mathbf{P} + \mathbf{P.new}$ 
      remove worst individuals from population(P) until  $size(\mathbf{P})$  reaches initial size  $N$ 
    end if
    if (DOP)
      replace  $size(\mathbf{P.new})$  worst individuals from population(P) by P.new
    end if
  end while

```

Procedure: **create new individuals**

```

matingEvents  $\leftarrow N/2$            /*  $N$  is the population size */
successfulMatings  $\leftarrow 0$ 
failedMatings  $\leftarrow 0$ 
while (successfulMatings < 1) do
  for ( $i \leftarrow 1$  to  $matingEvents$ ) do
    select two chromosomes ( $c_1, c_2$ )           /* Any method may be used here */
    compute Hamming distance  $H(c_1, c_2)$ 
    if ( $H(c_1, c_2) \geq T$ )
      crossover and mutate
      successfulMatings  $\leftarrow$  successfulMatings+1
    end if
    if ( $H(c_1, c_2) < T$ )
      failedMatings  $\leftarrow$  failedMatings+1
    end if
  end for
  if ( $failedMatings > successfulMatings$ ) T  $\leftarrow$  T-1
  else T  $\leftarrow$  T+1
end while

```

(1) In each time step, VDMGA builds an auxiliary pool of chromosomes, with parents and offspring, and then creates the new population by selecting the best chromosomes from the pool. This means that all newly created (and evaluated) individuals may be excluded from the population (considering the “worst” case scenario). Since the study on DOPs performed for this chapter assumes that changes are not detectable – and this is the most general assumption, since changes are not always detectable (Branke, 2002) –, all individuals in the population must be (re)evaluated in each generation, even if they have been created in a previous generation. Individuals with fitness values corresponding to previous shapes of the search space will mislead the search and modify performance metrics in a wrong

manner. (If changes were detectable, reevaluations would only be necessary when detecting a change.) Therefore, when dealing with DOPs, it is better to introduce a larger number of new individuals in VDMGA's population, not only to diminish reevaluations, but also to bring a larger amount of genetic material into the population. For that purpose, original VDMGA replacement strategy is substituted by the following process: all new individuals  $N'$  are introduced in the new population, replacing the worst  $N'$  old chromosomes – see pseudo-code for details.

(2) The original initial threshold value was set to  $L-1$ , such that a strong exploratory behavior is guaranteed to take place in the beginning of the search. Starting with  $L-1$ , results showed that VDMGA self-regulates the threshold in the first generation according to the conditions of the problem. The adaptive characteristic of the threshold and the robustness of VDMGA to its initial value suggested that it might be convenient to treat threshold's initial value as a constant and let the algorithm self-tune the parameter, thus reducing the complexity of the parameter's space. Since the expected ratio of dissimilar alleles in two random binary chromosomes is equal to 0.5 (considering infinite strings) it is likely that during the first generation ( $t = 1$ ) the threshold decreases to values around  $0.5 \times L$ . On the other hand, experimental results showed that the threshold value in the following generations depends on population size ( $N$ ) and length of the chromosome ( $L$ ): tests performed in (Fernandes & Rosa, 2008) show that the threshold value at the end of the first generation varies from 49.4% and 67.5% of the chromosome length  $L$  depending on  $N$  and  $L$ . As stated before, VDMGA needs to (re)evaluate all the old chromosomes in the population in order to deal with DOPs. If the algorithm passes through an initial stage, during which few new chromosomes are created, until it reaches a more stable threshold value, then a prohibitive number of reevaluations are performed, delaying the algorithm and compromising the first stage of optimization, especially when the changes occur fast. For that reason, initial threshold value is set to a lower value when the problem is dynamic. A value below  $0.5 \times L$  is sufficient. In the tests performed for this study and described in the following sections, initial threshold was set to  $0.25 \times L$ .

## 5. Performance and scalability on static environments

In (Fernandes & Rosa, 2008), VDMGA was subject to a wide range of experiments on some optimization functions frequently found in EAs literature. The test suite included unimodal and multimodal functions (with and without regular arrangement of local optima), a step function without local gradient information, scalable functions, high dimensional functions and complex combinatorial functions. VDMGA was compared with traditional GAs, CHC (Eschelmann, 1991) and nAMGA (Fernandes et al., 2000). pAMGA (Fernandes et al., 2000) was also included in the tests in order to compare analogous assortative and disassortative mating strategies and demonstrate that the former are more efficient in solving the proposed optimization problems. Overall results displayed VDMGA's superior performance when compared to other GAs (while statistically equivalent to nAMGA in some functions, VDMGA proved to be more efficient when facing the harder problems). Please refer to (Fernandes & Rosa, 2008) for a detailed description of the test set and results.

A simple scalability test is also provided in (Fernandes & Rosa, 2008). Using the 4-bit fully deceptive function (Whitley, 1991), results confirm the assumption that VDMGA's optimal population sizes are smaller than standard GA's. Consequently, the slope of the scalability log-log curve is reduced in VDMGA when compared with a generational GA and a steady-

state GA, even if only by a small amount. For this chapter, VDMGA's scalability is investigated in  $l$ -trap function. The main interest is to perceive how VDMGA reacts to increasing the number of  $l$ -traps that are juxtaposed and summed together.

### 5.1 Trap functions and VDMGA's scalability

To investigate how VDMGA's scales on landscapes with different characteristics, experiments were conducted with trap functions, which were used as subproblems to construct larger problems. A trap function is a piecewise-linear function defined on unitation (the number of ones in a binary string). There are two distinct regions in search space, one leading to a global optimum and the other leading to the local optimum (see figure 2). In general, a trap function is defined as in equation 1.

$$\text{trap}(u(\bar{x})) = \begin{cases} \frac{a}{z}(z - u(\bar{x})), & \text{if } u(\bar{x}) \leq z \\ \frac{b}{l-z}(u(\bar{x}) - z), & \text{otherwise} \end{cases} \quad (1)$$

where  $u(\bar{x})$  is the unitation function, defined as:

$$u(\bar{x}) = u(x_1, \dots, x_l) = x_1 + \dots + x_l = \sum_{i=0}^l x_i \quad (2)$$

and  $a$  is the local optimum,  $b$  is the global optimum,  $l$  is the problem size ( $l$ -bit trap function) and  $z$  is slope-change location separating the attraction basin of the two optima as depicted in figure 1.

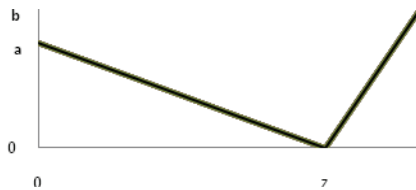


Fig. 1. Generalized  $l$ -trap function.

Depending on the parameter setting, trap functions may be deceptive or not. Deceptive problems are functions where low-order building-blocks do not combine to form higher order building-blocks. Instead, low-order building-blocks may mislead the search towards local optima, thus challenging GA's search mechanisms. For a trap function to be deceptive, the ratio  $r$  between the local ( $a$ ) and global ( $b$ ) optimum must be so that:

$$r \geq \frac{2 - \frac{1}{L-z}}{2 - \frac{1}{z}} \quad (3)$$

In the experiments, 2-bit, 3-bit and 4-bit trap functions were defined with the following parameters:  $a = l-1$ ;  $b = l$ ;  $z = l-1$ . This way, equation 1 may be simplified:

$$F(u(\vec{x})) = \begin{cases} l, & \text{if } u(\vec{x}) = l \\ l - 1 - u(\vec{x}), & \text{otherwise} \end{cases} \quad (4)$$

Please note that with these settings, the ratio  $r$  of the 2-trap function is below the deception threshold, while 4-trap is fully deceptive since the condition of equation 3 is satisfied. The ratio of the 3-trap function is equal to the threshold, which means that the function lies in the region between deceptive and non-deceptive. Under these conditions, it is possible to investigate not only how standard GAs and VDMGA scale on  $l$ -trap functions, but also to observe how that scaling varies when moving from non-deceptive to fully deceptive search spaces. For that purpose,  $L$ -bit decomposable functions were constructed by juxtaposing  $m$  trap functions and summing the fitness of each sub-function to obtain the total fitness:

$$f(\vec{x}) = \sum_{i=1}^m \text{trap}(\vec{x}_i) \quad (5)$$

For each trap and each size  $m$ , a standard generational GA (GGA) and VDMGA were run with several values of population size  $N$ . Starting from  $N = 4$ , optimal population size was determined by the bisection method (Sastry, 2001). The success rate (percentage of runs in which the global optimum was attained) and the average evaluations needed to find the solution (Average Evaluations to a Solution - AES) were measured. Each configuration was executed for 50 times and the results are averaged over those runs. The best configuration was defined as the one with 98% success rate and lower AES. Then, AES optimal population size values corresponding to the best run were plotted and the resulting log-log graphics are depicted in figure 2. The algorithms were tested with uniform crossover and no mutation. Crossover probability,  $p_c$ , was set to 1.0. Selection method is binary tournament ( $k_{ts} = 1.0$ ). (Please note that without mutation it is simply required that one bit is set to 0 or 1 in the entire population for the run to be declared not successful.)

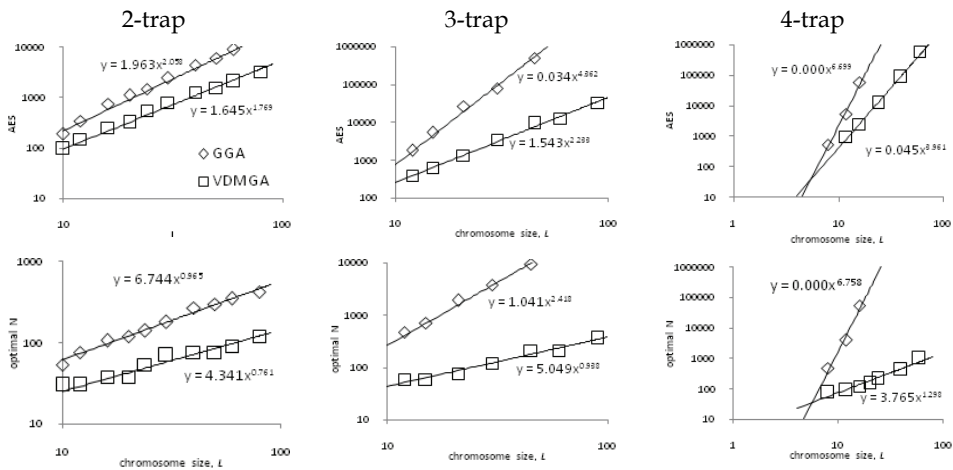


Fig. 2. Scalability with trap functions. Optimal population size and AES values for different problem size  $L = l \times m$ .

When solving 2-trap functions the algorithms behave similarly, but when the trap dimension increases to 3 and 4, the differences in the scalability is much more noticeable. The difficulty that deceptive trap functions pose to GAs is rather clear when noticing that GGA optimal population size values are close to search space size, that is,  $2^L$ , when solving 4-trap functions. VDMGA significantly reduces the slope of the scalability curve, revealing a good ability to maintain diversity and recombine information in order to achieve the higher order building-blocks. Since VDMGA maintains genetic diversity at a higher level, smaller populations are sufficient to find the global optimum, and thus fewer evaluations are required to converge to that same optimal solution.

VDMGA is a steady-state algorithm, and due to its structure, most of the generations keep the best solutions in the population. When compared to a GGA, which holds no elitism and the offspring completely replaces the parents' population, it is expected that it scales better. To avoid any misinterpretation of the results provided by VDMGA, another scalability test was performed to compare the algorithm with a GGA with 2-elitism (2-e), and a steady-state GA (SSGA) in which half of the population is replaced by the offspring (the worst chromosomes in the current population are replaced by  $N/2$  newly generated chromosomes). Results, presented in figure 3, show that both SSGA and GGA 2-e maintain a better scalability than GGA when raising the size of the trap from 2 to 4. In addition, SSGA keeps its performance very close to VDMGA when solving not only 2-traps, but also 3-traps.

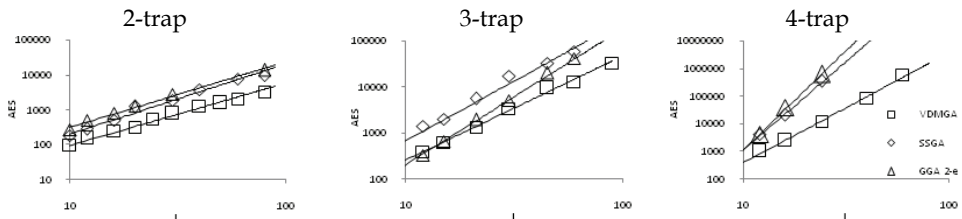


Fig. 3. Scalability with trap functions. Comparing VDMGA with an elitist generational GA (GGA 2-e) and a steady-state GA (SSGA).

However, when reaching 4-trap functions, it is clear that VDMGA scales better than elitist GGA and SSGA. It may be assumed now that this improved scalability is to a great extent due to the fact that VDMGA maintains a higher diversity during the run (Fernandes & Rosa, 2008), and not to its steady-state nature. Scalability tests are very important and useful because when tackling real-world problems, the algorithm may be requested to codify solutions in extremely large binary strings. If the GA does not scale well, optimization becomes practically impossible above a certain problem size. Scalability issues have been increasingly raising the interest of EAs research community, especially amongst EDAs (Pelikan, Goldberg & Lobo, 1999; Llorraña & Lozano, 2002) researchers.

## 6. VDMGA on dynamic optimization problems

The test environment proposed in (Yang & Xiao, 2005) was used to create an experimental setup for VDMGA on DOPs. Given a stationary problem  $f(x) (x \in \{0,1\}^L)$  where  $L$  is the chromosome length, the dynamic environments may be constructed by applying a binary mask  $M \in \{0,1\}^L$  to each solution before its evaluation in the following manner:

$$f(x, t) = f(x \text{ XOR } M(k)) \quad (6)$$

Where  $t$  is the generation index,  $k = \frac{t}{\tau}$  is the period index and  $f(x, t)$  is the fitness of solution  $x$ .  $M(k)$  can be incrementally generated as follows:

$$M(k) = M(k-1) \text{ XOR } T(k) \quad (7)$$

where  $T(k)$  is an intermediate binary mask for every period  $k$ . This mask  $T(k)$  has  $\rho \times L$  ones, where  $\rho$  is a value between 0 and 1.0 which controls the intensity or severity of change. Notice that  $\rho = 0$  corresponds to a stationary problem since  $T$  vectors will carry only 0's and no change will occur in the environment. On the other hand,  $\rho = 1$  will guarantee the highest degree of change, that is, for instance, if a solution to a problem is a vector of 1's, then the dynamic solution will oscillate between a vector of 1's and a vector of 0's. Therefore, by changing  $\rho$  and  $\tau$  in the previous set of equations it is possible to control two of the most important features when testing algorithms on DOPs: severity ( $\rho$ ) and speed ( $\tau$ ) of change (Angeline, 1997).

The generator was applied to trap functions. GGA 2-e and SSGA were tested in order to compare them with VDMGA. It is not the aim of this study to compare VDMGA with the best GAs in solving DOPs, even because there is hardly any evidence of a GA that consistently outperforms any other in a wide range of problems and dynamics. Instead, the study of the effects of VDMGA's diversity maintenance on its behavior on dynamic environments is the main aim of this section: the way dissortative mating may be used in order to improve GAs performance and on which kind of DOPs that improvement is more noticeable. Nevertheless, a commonly used algorithm on dynamic optimization studies was added to the test bench: RIGA (see section 3). Two variations were tested. In RIGA 1, the immigrants replace randomly selected individuals from the population, while in RIGA 2 the  $r_r$  immigrants replace the worst  $r_r$  individuals in the population. (Both RIGA were implemented with 2-elitism. Non-elitist GAs were tested but the performance on trap DOPs was very poor. VDMGA outperformed non-elitist GAs on every problem and ( $\rho$ ,  $\tau$ ) configuration, but such a test is clearly unfair to standard and Random Immigrants GAs. All the algorithms were tested with  $N = 240$ ,  $p_c = 1.0$ , uniform crossover and binary tournament ( $k_{is} = 1$ ). Performance was measured by comparing the mean *best\_of\_generation*:

$$\text{mean best\_of\_generation} = \frac{\sum_{i=0}^R \frac{\sum_{j=0}^T \text{bestfitness}_{i,j}}{T}}{R} \quad (8)$$

where  $T$  is the number of generations and  $R$  is the number of runs (30 in all the experiments). Several tests were conducted by varying severity ( $\rho$ ) and speed ( $\tau$ ) of change:  $\rho$  was set to 0.05, 0.6 and 0.95; speed of change  $\tau$  was set to 10, 100 and 200 generations. This means that 9 kinds of environmental changes were tested for each function and algorithm. Every environment was tested with 10 periods of change, thus making  $T = 100$  for  $\tau = 10$ ,  $T = 1000$  for  $\tau = 100$  and  $T = 2000$  for  $\tau = 200$ . Since it is expected that the

optimal mutation rate is not equal for every GA in the test bench, it is of extreme importance to test the algorithms with different  $p_m$ . Values ranged from  $0.5/L$  to  $5/L$  and the results displayed on tables 1-3 correspond to best configurations (best configurations were determined by averaging the nine performance values). In order to properly compare the algorithms it is imperative that each GA performs the same number of function evaluations in each generation. Otherwise, during each period between changes, different GAs may be requiring different computation effort. For that reason, RIGAs population size must be set to  $N-r_r$ , because RIGA performs extra  $r_r$  evaluation in each time step. For RIGA's tests in this section,  $r_r$  was set to 24, therefore,  $N$  is equal to 216. In addition, since this study assumes that changes in the environment are not detectable, all chromosomes must be evaluated in each generation, even those that have already been evaluated in a previous generation, as in VDMGA (please remember that VDMGA is a steady-state algorithm, that is, parents and children may belong to the same population). For the same reason, SSGA also reevaluates the fraction (half) of the population that has not been replaced by children. (GGA, due to its 2-elitism, must also reevaluate, in each generation, the two best chromosomes in the population.) This way, VDMGA always performs  $N$  fitness calculations in each generation but only a fraction of those evaluations are performed on new individuals. This feature is expected to penalize VDMGA's performance on DOPs with very fast changes (low  $\tau$ ), since it may happen that for some periods of time only a small amount of new genetic material (new individuals) are inserted in the population in each generation. Actually, this outcome is confirmed on the first test, performed on 3-trap functions.

Table 1 shows the results obtained by the various GAs on a function constructed by juxtaposing ten 3-trap subfunctions ( $L = 30$ ). A statistical comparison was carried out by  $t$ -tests with 58 degrees of freedom at a 0.05 level of significance. The (+) signs means that the corresponding algorithm is significantly better than VDMGA on that particular configuration of  $\rho$  and  $\tau$ . A (~) sign means that the performance is statistically equivalent and (-) sign means that the GA performs worst than VDMGA. A general observation of table 1 shows that only when  $\tau = 10$  the GAs consistently outperform VDMGA. With lower speed, VDMGA has always a better performance than the other algorithms in the test bench when  $\rho = 0.6$  and  $\rho = 0.95$ , being statistically equivalent when  $\rho = 0.05$ . This was an expected outcome, due to what was stated above about VDMGA's ration between function evaluations in each time step and new chromosomes inserted in the population.

Table 2 shows the results with 4-trap functions ( $L = 12$ ). Values appear to be more balanced in this case: all the algorithms perform similarly, but when increasing the size of the problem to  $L = 24$  - see table 3 -, VDMGA improves its performance when compared to other GAs when  $\tau = 100$  and  $\tau = 200$  (please remember that VDMGA is expected to face some difficulties when facing fast changing environments. However, the algorithm performs well in 12-bit and 24-bit 4-trap function when  $\rho = 0.05$  and  $\tau = 10$ .)

An unexpected result occurs when speed of change is slow and  $\rho = 0.95$ . For instance, when  $\tau = 200$ , RIGAs outperforms VDMGA. But when looking at the dynamic behavior of the algorithms, in figure 4, a possible explanation arises for this particular result. Figure 4 shows the dynamics of VDMGA, SSGA and RIGA 2 when tracking the extrema of 4-trap functions ( $L = 24$ ) by plotting the *best\_of\_generation* values over all generations. When  $\rho = 0.6$  and  $\tau = 200$  the graphs shows that VDMGA becomes closer to the optimum (and results on table 3 confirm that VDMGA outperforms other algorithms). However, when increasing  $\rho$  to 0.95,

$\tau$ $\rho$	3-trap ( $L = 30$ )				
	GGA ( $p_m = 1/L$ )	SSGA ( $p_m = 1/2L$ )	RIGA 1 ( $p_m = 1/L$ )	RIGA 2 ( $p_m = 1/L$ )	VDMGA ( $p_m = 1/L$ )
10	26.06	26.020	25.938	26.144	<b>25.319</b>
0.05	$\pm 0.978$ (+)	$\pm 0.506$ (+)	$\pm 0.661$ (+)	$\pm 0.819$ (+)	$\pm 0.556$
10	22.078	21.467	21.934	21.952	<b>21.227</b>
0.60	$\pm 0.266$ (+)	$\pm 0.289$ (+)	$\pm 0.305$ (+)	$\pm 0.362$ (+)	$\pm 0.280$
10	23.937	23.638	23.832	23.978	<b>22.877</b>
0.95	$\pm 0.278$ (+)	$\pm 0.326$ (+)	$\pm 0.221$ (+)	$\pm 0.237$ (+)	$\pm 0.404$
100	29.712	29.656	29.674	29.664	<b>29.622</b>
0.05	$\pm 0.090$ (~)	$\pm 0.082$ (~)	$\pm 0.145$ (~)	$\pm 0.175$ (~)	$\pm 0.103$
100	26.293	26.095	26.322	26.258	<b>26.444</b>
0.60	$\pm 0.186$ (-)	$\pm 0.257$ (-)	$\pm 0.292$ (-)	$\pm 0.300$ (-)	$\pm 0.250$
100	25.605	25.730	25.628	25.597	<b>25.999</b>
0.95	$\pm 0.137$ (-)	$\pm 0.098$ (-)	$\pm 0.163$ (-)	$\pm 0.121$ (-)	$\pm 0.242$
200	29.851	29.822	29.849	29.852	<b>29.821</b>
0.05	$\pm 0.051$ (~)	$\pm 0.075$ (~)	$\pm 0.526$ (~)	$\pm 0.056$ (~)	$\pm 0.050$
200	27.120	26.972	27.350	27.314	<b>27.826</b>
0.60	$\pm 0.200$ (-)	$\pm 0.261$ (-)	$\pm 0.225$ (-)	$\pm 0.272$ (-)	$\pm 0.170$
200	25.838	25.978	25.802	25.818	<b>26.211</b>
0.95	$\pm 0.129$ (-)	$\pm 0.096$ (-)	$\pm 0.122$ (-)	$\pm 0.180$ (-)	$\pm 0.185$

Table 1. Results on 3-traps ( $L = 30$ ). Mean *best\_of\_generation* and corresponding standard deviation values (results averaged over 30 runs).

$\tau$ $\rho$	4-trap ( $L = 12$ )				
	GGA ( $p_m = 3/L$ )	SSGA ( $p_m = 3/L$ )	RIGA 1 ( $p_m = 4/L$ )	RIGA 2 ( $p_m = 4/L$ )	VDMGA ( $p_m = 2/L$ )
10	10.712	11.307	10.800	10.782	<b>11.283</b>
0.05	$\pm 0.215$ (-)	$\pm 0.271$ (~)	$\pm 0.171$ (-)	$\pm 0.162$ (-)	$\pm 0.254$
10	10.800	10.484	10.783	10.833	<b>10.585</b>
0.60	$\pm 0.177$ (+)	$\pm 0.176$ (~)	$\pm 0.178$ (+)	$\pm 0.179$ (+)	$\pm 0.175$
10	10.914	11.360	10.798	10.825	<b>11.436</b>
0.95	$\pm 0.213$ (-)	$\pm 0.193$ (~)	$\pm 0.174$ (+)	$\pm 0.191$ (+)	$\pm 0.204$
100	11.653	11.948	11.700	11.705	<b>11.957</b>
0.05	$\pm 0.109$ (-)	$\pm 0.031$ (~)	$\pm 0.010$ (-)	$\pm 0.088$ (-)	$\pm 0.020$
100	11.687	11.661	11.713	11.725	<b>11.672</b>
0.60	$\pm 0.089$ (~)	$\pm 0.074$ (~)	$\pm 0.020$ (~)	$\pm 0.075$ (~)	$\pm 0.060$
100	11.710	11.622	11.735	11.688	<b>11.696</b>
0.95	$\pm 0.080$ (~)	$\pm 0.076$ (-)	$\pm 0.016$ (~)	$\pm 0.068$ (~)	$\pm 0.062$
200	11.823	11.981	11.842	11.843	<b>11.981</b>
0.05	$\pm 0.066$ (-)	$\pm 0.010$ (~)	$\pm 0.012$ (-)	$\pm 0.034$ (-)	$\pm 0.012$
200	11.842	11.823	11.847	11.873	<b>11.831</b>
0.60	$\pm 0.051$ (~)	$\pm 0.027$ (~)	$\pm 0.017$ (~)	$\pm 0.035$ (+)	$\pm 0.036$
200	11.852	11.691	11.863	11.864	<b>11.742</b>
0.95	$\pm 0.049$ (+)	$\pm 0.055$ (-)	$\pm 0.014$ (+)	$\pm 0.037$ (+)	$\pm 0.028$

Table 2. Results on 4-traps ( $L = 12$ ). Mean *best\_of\_generation* and standard deviation values, averaged over 30 runs.

$\tau$ $\rho$	4-trap ( $L = 24$ )				
	SGA ( $p_m = 1/L$ )	SSGA ( $p_m = 2/L$ )	RIGA 1 ( $p_m = 1/L$ )	RIGA 2 ( $p_m = 1/L$ )	VDMGA ( $p_m = 3/L$ )
10	18.394	19.710	18.301	18.417	<b>19.544</b>
0.05	$\pm 0.355$ (-)	$\pm 0.648$ (~)	$\pm 0.340$ (-)	$\pm 0.391$ (-)	<b><math>\pm 0.345</math></b>
10	18.270	17.777	18.142	18.066	<b>17.703</b>
0.60	$\pm 0.185$ (+)	$\pm 0.311$ (~)	$\pm 0.282$ (+)	$\pm 0.282$ (+)	<b><math>\pm 0.289</math></b>
10	20.807	20.489	20.682	20.747	<b>19.724</b>
0.95	$\pm 0.200$ (+)	$\pm 0.347$ (+)	$\pm 0.230$ (+)	$\pm 0.161$ (+)	<b><math>\pm 0.284</math></b>
100	19.136	22.370	19.091	19.125	<b>23.421</b>
0.05	$\pm 0.408$ (-)	$\pm 0.629$ (-)	$\pm 0.434$ (-)	$\pm 0.583$ (-)	<b><math>\pm 0.141</math></b>
100	20.570	20.630	20.474	20.593	<b>20.518</b>
0.60	$\pm 0.242$ (~)	$\pm 0.293$ (~)	$\pm 0.233$ (~)	$\pm 0.274$ (~)	<b><math>\pm 0.323</math></b>
100	21.465	21.308	21.418	21.452	<b>21.472</b>
0.95	$\pm 0.099$ (~)	$\pm 0.103$ (-)	$\pm 0.116$ (~)	$\pm 0.076$ (~)	<b><math>\pm 0.176</math></b>
200	19.065	23.385	19.220	19.430	<b>23.746</b>
0.05	$\pm 0.140$ (-)	$\pm 0.347$ (-)	$\pm 0.760$ (-)	$\pm 0.895$ (-)	<b><math>\pm 0.0726</math></b>
200	20.947	21.058	20.851	20.800	<b>21.670</b>
0.60	$\pm 0.267$ (-)	$\pm 0.228$ (-)	$\pm 0.294$ (-)	$\pm 0.231$ (-)	<b><math>\pm 0.175</math></b>
200	21.509	21.332	21.503	21.495	<b>21.354</b>
0.95	$\pm 0.065$ (+)	$\pm 0.136$ (~)	$\pm 0.087$ (+)	$\pm 0.082$ (+)	<b><math>\pm 0.166</math></b>

Table 3. Results on 4-traps. Mean *best\_of\_generation* and standard deviation values, averaged over 30 runs.

the curves become much different. The shape of RIGA's curve may be easily explained by the characteristics of the trap functions used in this study: the global optimum of the functions is the string with all 1's and the local optimum is the string with all 0's. RIGA is stuck in a region of the search space and, when the environment changes dramatically ( $\rho = 0.95$ ), what were once chromosomes near the global optimum local then become (nearly) local optimum solutions. With 4-trap functions and  $L = 24$ , global optimum is 24 and local optimum value is 18. Please note how RIGA oscillates between values near 24 and 18. The algorithm is not able to track the optimum; it just "waits", for the global optimum to pass by every other period of change. Exclusively looking at mean *best\_of\_generation* values may conduce to a misinterpretation of GAs abilities to solve DOPs.

Another aspect is worth notice. It is clear that RIGA 2 is not able to track the optima when changes are small ( $\rho = 0.05$ ), at least not as able as VDMGA and SSGA. Random material inserted in the population is not an appropriate strategy to deal with an environment that shifts only by a small amount. For  $\rho = 0.05$  and low speed of change ( $\tau = 100$  and  $\tau = 200$ ) VDMGA tracks the optima with much more ability than SSGA, even if it is slower in the first stage of search: only three periods of  $\tau$  generations are needed for VDMGA to track the optima and remain close to it in the following periods. RIGA 2 appeared to perform well on 24-bit 4-trap when compared to other algorithms (table 3). However, a closer inspection, by plotting the evolution of the tracking process, reveals that the algorithm fails when changes are both small ( $\rho = 0.05$ ) and severe ( $\rho = 0.95$ ). VDMGA, on the other hand, maintains a more stable performance trough all the different combinations of speed and severity of change, being particular able to track the optimum when  $\rho = 0.05$ .

### 7. Genetic diversity and threshold dynamics

As described above, assortative and dissortative mating have effects on the frequency of heterozygous and homozygous genotypes. Consequently, population diversity may also be affected: dissortative tends to increase genetic diversity while assortative decreases it. This may also be true when dealing with artificial systems such as GAs. Previous reports

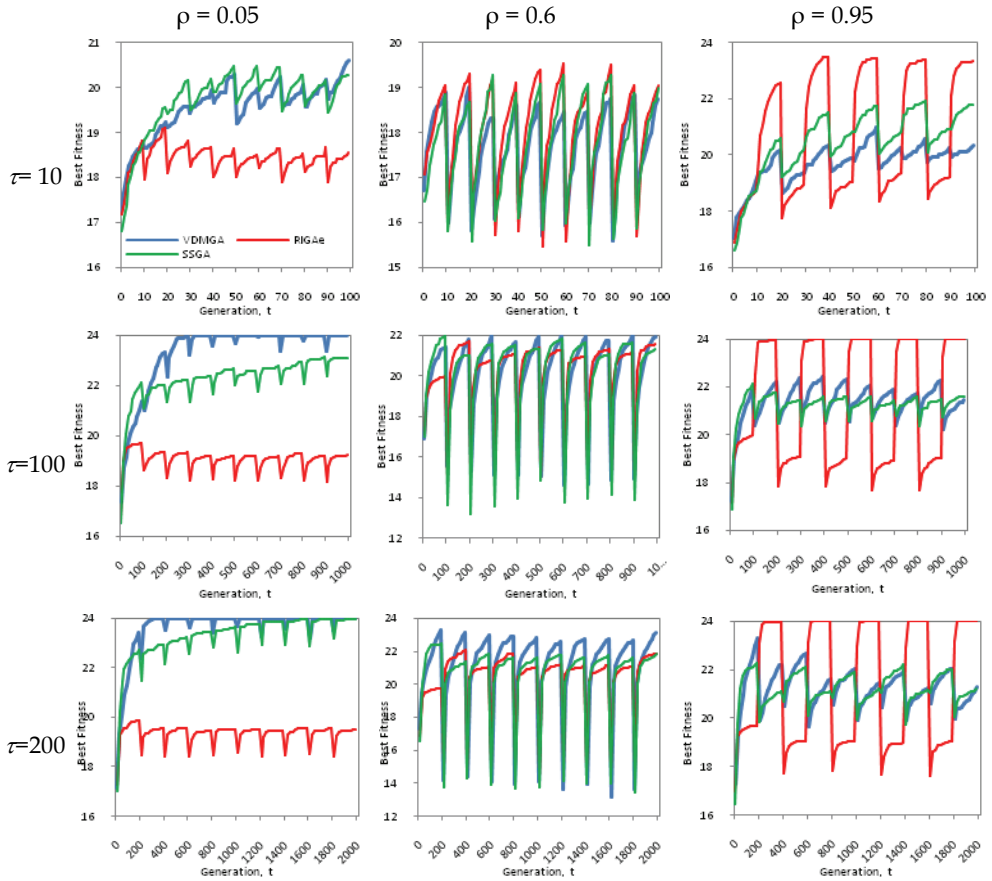


Fig. 4. Dynamics when tracking 4-trap functions ( $L = 24$ ). *Best\_of\_generation* curves.

(Fernandes & Rosa, 2001; Fernandes, 2002) show that the variation of diversity in GAs populations is influenced by the chosen mating strategy. In (Fernandes & Rosa, 2008), a study on genetic diversity also confirmed this assumption. To measure diversity, the following equation was used:

$$d(P) = \frac{\sum_{i=1}^L \min(F_i, 1 - F_i)}{L / 2} \tag{9}$$

where  $F_i = \sum_{j=1}^N P_i(j)$  and  $P_i(j) = \begin{cases} 1, & \text{if } j^{\text{th}} \text{ gene of chromosome } i \text{ has allele } 1 \\ 0, & \text{if } j^{\text{th}} \text{ gene of chromosome } i \text{ has allele } 0 \end{cases}$

Diversity was inspected on VDMGA, SSGA and RIGA 2. For that purpose, a problem with ten 4-trap subfunctions was used ( $L = l \times m = 4 \times 10 = 40$ ). Population size was set to 100,  $p_c$  was set to 1.0 (binary tournament selection and uniform crossover) and different mutation rates  $p_m$  were tested. The algorithms were run for 100 generations. Each run was repeated for 30 times and the results are the average over those runs.

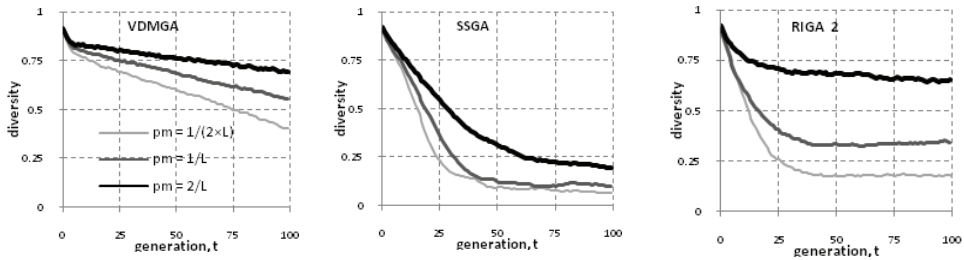


Fig. 5. Genetic diversity.

Graphics in figure 5 show similar results as in previous reports: VDMGA maintains a higher diversity than SSGA, even when comparing different mutation rates. RIGA gets closer to VDMGA’s diversity but, as depicted in figure 6, performance is much lower. By observing the growth of the best fitness in the population, it is clear that RIGA 2 is outperformed by SSGA, converging to a lower local optimum. On the other hand, VDMGA, although being slower in a first stage of search, attains higher fitness values, which are still growing when  $t = 100$ . These results illustrate how important are the genetic diversity maintenance schemes, and not diversity maintenance itself. RIGA, although maintaining the diversity for a longer period, is outperformed by SSGA on this particular test.

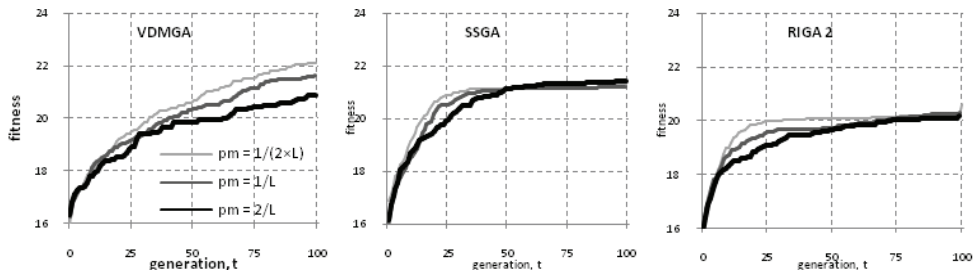


Fig. 6. Best fitness on 4-traps ( $L = 24$ ).

A final test was conducted with the aim of investigating diversity when the environment changes. For that purpose, a 4-trap DOP with  $L = 4$  was used. GAs parameters were set as in previous experiment. VDMGA’s diversity is compared with SSGA in figure 7 (only five periods of change are shown in the graphs), for two configurations of  $(\rho, \tau)$ . As expected,

VDMGA maintains a higher diversity throughout the successive search periods, even with lower mutation rates.

VDMGA's threshold values during a run of each one of the previous experiments ( $\rho = 0.05$  and  $\tau = 100$ ;  $\rho = 0.6$  and  $\tau = 200$ ) may be seen in figure 8 (with  $p_m = 1/L$ ). The graphics indicate that the threshold reacts to the changes in the environment when it is close to 0: when the environment shifts, the threshold tends to increase. The explanation for this outcome is simple and resides in the fact that after a change occurs, new genetic material enters in a previously converged population, allowing the threshold to increase because successful matings have also increase. An amplified threshold will then prevent mating between similar individuals and continue to guarantee higher genetic diversity.

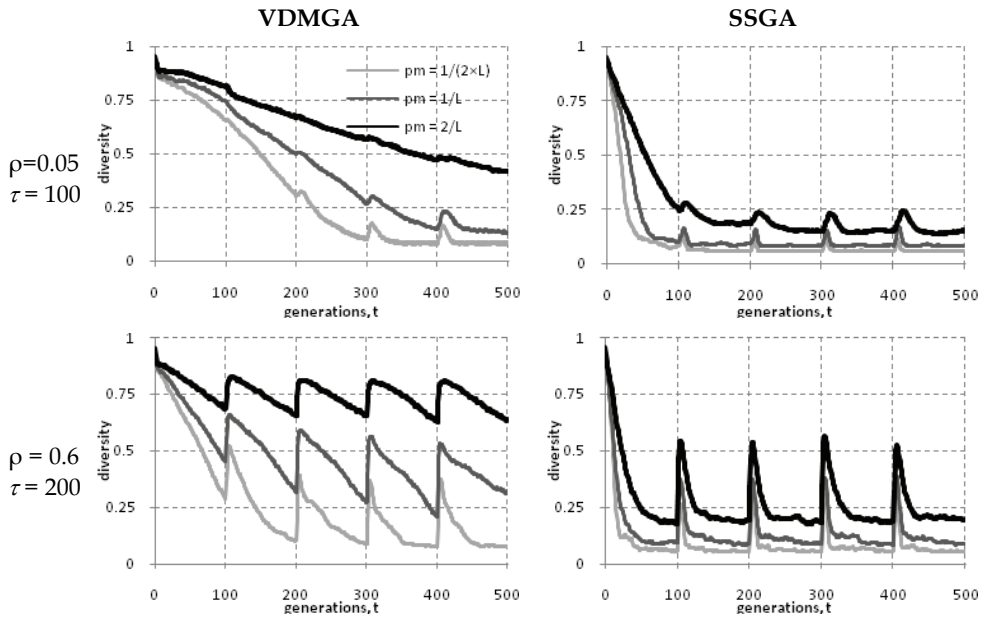


Fig. 7. SSGA and VDMGA's diversity on dynamic 4-trap functions.

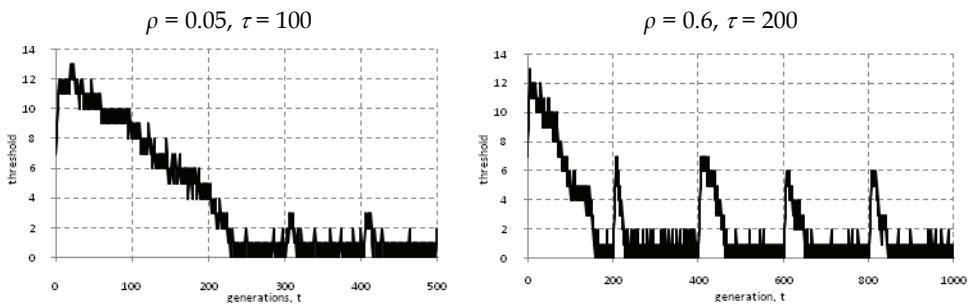


Fig. 8. VDMGA's threshold value. Mutation rate,  $p_m = 1/L$

## 8. Conclusions

This chapter presented a study on Genetic Algorithms (GA) with dissortative mating. A survey on non-random mating was given, in which the most prominent techniques in Evolutionary Computation literature were presented and described. In addition, a survey on Bio-inspired Computation applied to Dynamic Optimization Problems (DOPs) was also given, since DOPs was one of the main aims of the experimental study performed for this chapter. The experiments were performed with the aim of checking the ability of Variable Dissortative Mating GA (VDMGA) on tracking the extrema in dynamic problems. VDMGA, presented in a recent work (Fernandes & Rosa, 2008), inhibits crossover when the Hamming distance between the chromosomes is below a threshold value. The threshold is updated (incremented or decremented) by a simple rule which is indirectly influenced by the genetic diversity of the population: it tends to decrease when the amount of successful crossovers is superior to the number of failed attempts in a generation; when the ratio of successful recombination events rises, the threshold will have a tendency to increase. VDMGA holds this mechanism without the need for further parameters than traditional GAs. In fact, the parameters that need to be tuned are reduced to population size and mutation rate. In addition, no replacement strategy has to be chosen: VDMGA is a steady-state GA in which the number of new chromosomes entering the population in each generation is controlled by the threshold value, genetic diversity and population's stage of convergence.

Scalability tests were performed in order to investigate how VDMGA reacts to growing problem size. Deceptive and non-deceptive trap functions were used for that purpose. The algorithm was tested and compared with traditional GAs. Results showed that VDMGA scales clearly better than other traditional GAs when the trap function is deceptive.

DOPs experiments demonstrated that in most of the cases, VDMGA is able to perform equally or better than other GAs, except when the speed of change is high. In particular, VDMGA outperformed, in general, the Random Immigrants GA, which a typical algorithm used in DOPs studies to compare other methods performance. Statistical t-tests were performed, giving stronger reliability to the conclusions.

A study on the genetic diversity was also performed. As expected, VDMGA maintains a higher diversity throughout the run. The speed of the algorithm may be reduced in a first stage of search (and that is one of the reasons VDMGA is not so able to solve fast DOPs), but the diversity of its population gives it the ability to converge more often to the global optimum.

VDMGA is a simple yet effective algorithm to deal with static and dynamic environments. It holds no more parameters than a standard GA. When regarding DOPs, VDMGA may be classified in the category of methods that preserve diversity in order to tackle DOPs (see section 3). Thus, it avoids the complexity of methods that hold memory schemes (which in general need rules and parameters to determine how to deal with memory), and the lower range of problems in which algorithms that react to changes may be applied. Changes in DOPs are not always detectable and a reaction to changes assumes that it is possible to detect when the environment shifts.

## 9. Acknowledgments

First author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH/BD/18868/2004, partially supported by *Fundação para a Ciência e a Tecnologia (ISR/IST* plurianual funding) through POS\_Conhecimento Program that includes FEDER funds.

## 10. References

- Angeline, P. (1997). Tracking Extrema in Dynamic Environments. *Proceedings of the 6th International Conference on Evolutionary Programming*, Springer, pp. 335-345.
- Arabas, J.; Michalewicz, Z.; Mulawka, J. (1994). GAVaPS - A genetic algorithm with varying population size, *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, IEEE, Vol. 1: pp. 73-78.
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, Oxford University, New York.
- Bak, P.; Tang, C.; K. Wiesenfeld, K. (1987). Self-organized criticality: an explanation of  $1/f$  noise. *Physical Review of Letters*, vol. 59, pp. 381-384.
- Bak, P.; K. Sneppen (1993). Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review of Letters*, vol. 71, pp. 4083-4086.
- Baluja, S. (1994). Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, *Technical Report CMU-CS-94-163*, Carnegie Mellon University, USA.
- Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. *Proceedings of the 1999 Congress on Evolutionary Computation*, IEEE, pp. 1875-1882.
- Branke, J. (2002). *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers.
- Branke, J.; Schmeck, H. (2002), Designing evolutionary algorithms for dynamic optimization problems. *Theory and Application of Evolutionary Computation: Recent Trends*, A. Ghosh and S. Tsutsui (editors), pp. 239-262.
- Cobb, H.G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, *Technical Report AIC-90-001*, Naval Research Laboratory, Washington, USA.
- Craighurst R, Martin W (1995) Enhancing GA performance through crossover prohibitions based on ancestry, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufman, pp. 130-135.
- De, S.; Pal, S.K.; Ghosh, A. (1998). Genotypic and phenotypic assortative mating in genetic algorithm. *Information Science* 105: pp. 209-225.
- Eschelman, L.J. (1991). The CHC algorithm: How to have safe search when engaging in non-traditional genetic recombination, *Proceedings of Foundations of Genetic Algorithms*, Academic Press, 1: pp. 70-79.
- Eschelman, L.J.; Schaffer, J.D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. *Proceedings of the fourth International Conference on Genetic Algorithms*, Morgan Kaufman, pp. 115-122.
- Fernandes, C.M.; Tavares, R.; Rosa, A.C. (2000). NiGAVaPS - Outbreeding in genetic algorithms, *Proceedings of 2000 Symposium on Applied Computing*, ACM, pp. 477-482.
- Fernandes, C.M.; Tavares, T.; Munteanu, C.; Rosa, A.C. (2001). Using Assortative Mating in Genetic Algorithms for Vector Quantization Problems, *Proceedings of 2001 Symposium on Applied Computing*, ACM, pp. 361-365.
- Fernandes, C.M.; Rosa, A.C. (2001). A Study on Non-Random Mating in Evolutionary Algorithms Using a Royal Road Function. *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE, pp. 60-66.
- Fernandes, C.M. (2002) *Algoritmos Genéticos e Acasalamento não-aleatório*, Msc dissertation thesis, IST, Universidade Técnica de Lisboa, in Portuguese.

- Fernandes, C.M., Rosa, A.C. (2006). Self-Regulated Population Size in Evolutionary Algorithms, *Proceedings of 9<sup>th</sup> International Conference on Parallel Problem Solving from Nature*, LNCS 4193, 920-929.
- Fernandes, C.M.; Rosa, A.C.; Ramos, V. (2007). Binary ant algorithm. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, ACM, pp. 41-48.
- Fernandes, C.; Rosa, A.C. (2008). Self-adjusting the intensity of dissortative mating of genetic algorithms, *Journal of Soft Computing*, in press.
- Fernandes, C.; Merelo J.J.; Ramos, V.; Rosa, A.C. (2008a). A self-organized criticality mutation operator for dynamic optimization problems. to appear in *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, ACM.
- Fernandes C, Lima C, Rosa AC (2008b), UMDAs for Dynamic Optimization. to appear in *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, ACM Press.
- García-Martínez, C.; Lozano, M.; Molina, D. (2006). A Local Genetic Algorithm for Binary-Coded problems, In T. Runarsson et al. (eds.), *Proceedings of 9<sup>th</sup> International Conference on Parallel Problem Solving from Nature*, LNCS 4193, 192-201.
- García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez AM (2008). Global and local real-coded genetic algorithms based on parent-centric crossover operators, *European Journal of Operational Research*, 185(3): pp. 1088-1113.
- Glover, F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, 5: pp. 533-549.
- Grefenstette, J.J. (1992). Genetic algorithms for changing environments, *Proceedings of Parallel Problem Solving from Nature II*, North-Holland, pp. 137-144.
- Goldberg, D.E.; Smith, R.E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy, *Proceedings of the 2<sup>nd</sup> International Conference on Genetic Algorithms*, ACM, pp. 59-68.
- Guntsch, M.; Middendorf, M. (2002). Applying population based ACO to dynamic optimization problems. *Proceedings of 3<sup>rd</sup> International Workshop ANTS 2002*, Springer, pp. 111-122.
- Harik, G. R. (1999). Linkage learning via probabilistic modeling in the ECGA. *IlligAL Report No. 99010*, Illinois Genetic Algorithms Laboratory.
- Hillis, W. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure, *Artificial Life II*, Addison-Wesley, pp. 313-324.
- Huang, C.; J. Kaur, A.; Maguitman, L.; Rocha, L. (2007). Agent-based model of genotype editing, *Evolutionary Computation*, MIT Press, pp. 253-289.
- Jaffe, K. (1999). On the adaptive value of some mate selection techniques, *Acta Biotheoretica*, 47: pp. 29-40.
- Lorraña, P.; Lozano, J.A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Boston: Kluwer Academic Publishers, Boston.
- Louis, S.J.; Xu, Z. (1996). Genetic Algorithms for open shop scheduling and rescheduling. *Proceedings of the 11<sup>th</sup> International Conference on Computers and their Applications*, pp. 99-102.
- Lozano, M.; Herrera, F.; Krasnogor, N.; Molina, D. (2004). Real coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation Journal*, 12(3): pp. 273-302.
- Matsui K (1999) New selection method to improve the population diversity in genetic algorithms, In: *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*.
- Mauldin, M. (1984). Maintaining genetic diversity in genetic search, *National Conference on Artificial Intelligence*, AAAI, pp. 247-250.

- Mitchell, M. (1994). When will a GA outperform hillclimbing? *Advances in Neural Information Processing Systems*, 6: pp. 51-58.
- Ochoa, G.; Madler-Kron, C.; Rodriguez, R.; Jaffe, K. (1999). On sex, selection and the Red Queen. *Journal of Theoretical Biology* 199: pp. 1-9.
- Ochoa, G.; Madler-Kron, C.; Rodriguez, R.; Jaffe, K. (2005). Assortative mating in genetic algorithms for dynamic problems. *Proceedings of the 2005 EvoWorkshops*, LNCS 3449, pp. 617-622.
- Ochoa, G. (2006). Error Thresholds in Genetic Algorithms. *Evolutionary Computation*, 14(2): pp. 157-182.
- Ochoa, G.; Jaffe, K. (2006). Assortative Mating Drastically Alters the Magnitude of Error Thresholds. *Proceedings of 9<sup>th</sup> International Conference on Parallel Problem Solving from Nature*, LNCS 4193, pp. 890-899.
- Pelikan, M.; Goldberg D.; Lobo, F. (1999). A Survey of Optimization by Building and Using Probabilistic Models. *Technical Report 99018*, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory (IlligAL), IL, USA.
- Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.
- Petrowski, A. (1997). A new selection operator dedicated to speciation, *Proceedings of the 7<sup>th</sup> International Conference on Genetic Algorithms*, Morgan Kaufman, pp. 144-151.
- Ramos, V.; Fernandes, C.; Rosa, A.C. (2005). On self-regulated swarms, societal memory, speed and dynamics. *Proceedings of ALifeX*, MIT Press, pp. 393-399.
- Ronald, E. (1995). When selection meets seduction. *Proceedings of the 6<sup>th</sup> International Conference on Genetic Algorithms*, Morgan Kaufman, pp. 167-173.
- Roughgarden, J. (1979). *Theory of population genetics and evolutionary ecology*, Prentice-Hall.
- Russel, P.J. (1998). *Genetics*. Benjamin/Cummings.
- Ting, C; Sheng-Tu, L.; Chungnan, L. (2003). On the harmounious mating strategy through tabu search. *Journal of Information Sciences*, 156(3-4): pp. 189-214.
- Tinós, R; Yang, S. (2007). A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, 8: pp. 255-286.
- Todd, P.M.; Miller, G.F. (1991). On the sympatric origin of species: Mercurian mating in the quicksilver model. *Proceedings of the IV International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 547-554.
- Wagner, S.; Affenzeller, M. (2005). SexualGA: Gender-specific selection for genetic algorithms. *Proceedings of the 9th World Multiconference on Systemics, Cybernetics and Informatics*, vol.4, pp. 76-81.
- Whitley, D. (1988). GENITOR: a different genetic algorithm. *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, pp. 118-130.
- Whitley, D. (1991). Fundamental principles of deception in genetic search. *Foundations of Genetic Algorithms*, 1: pp. 221-241.
- Yang, S.; Yao, X. (2005). Experimental Study on population-based incremental learning algorithms for dynamic optimization problems. *Journal of Soft Computing*, 9(11): pp. 815-834.
- Yang, S. (2005). Memory-enhanced univariate marginal distribution algorithms. *Proceedings of the 2005 Congress on Evolutionary Computation*, ACM, pp. 2560-2567.
- Zlochin, M.; Birattari, M.; Meuleau, N.; Dorigo, M. (2004). Modelbased search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131: pp. 373-395.